**SpringMVC+Mysql实例**

首先我们先来了解一下什么是模式，模式就是解决某一类问题的方法论，把解决这类问题的解决方法归总到理论的高度，这就是模式。模式是一种指导，在一个良好的指导下，有助于开发人员完成任务。做出一个优秀的设计方案，能达到事半功倍的效果。而且会得到解决问题的最佳办法。

　　mvc模式起源于Smalltalk语言，mvc 是Model-View-Controller的简写。mvc减弱了业务逻辑接口和数据接口之间的耦合。使用MVC模式的好处有很多，可靠性强，高重用和可适应性，较低的生命周期成本，快速的部署，可维护性强等。里面的细节在这儿就不作过多的讲解。

SpringMVC的特点：

1、清晰的角色划分，Spring在Model、View和Controller方面提供了一个非常清晰的划分，这3个方面真正是各司其职，各负其责。

2、灵活的配置功能，因为Spring的核心是IOC,同样在实现MVC上，也可以把各种类当做Bean来通过XML进行配置。

3、提供了大量的控制器接口和实现类，这样开发人员可以使用Spring提供的控制器实现类，也可以自己实现控制器接口。

4、SpringMVC是真正的View层实现无关的，它不会强制开发员使用JSP,我们可以使用其他View技术，比如Velocity，Xskt等。

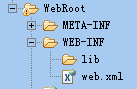
5、国际化支持，Spring的ApplicationContext提供了对国际化的支持，在这里可以很方便的使用。

6、面向接口编程，其实这不仅是springMVC的特点，整个Spring来看，这个特点都是很明显的，因为它使开发人员对程序易于进行测试，并且很方便的进行管理。

7、Spring提供了Web应用开发的一整套流程，而不仅仅是MVC，他们之间可以很方便的结合在一起。下面有一个自己做得例子，做完这个例子后真的体会到了SpringMVC的强大。

下面开始配置我们的Springmvc工程：

首先我们配置WEB-INF目录下的web.xml:



<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<servlet>

<!--springmvc的核心是DispatcherServlet，它负责控制整个页面的请求路径-->

<servlet-name>dispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!--初始化参数 >/WEB-INF/classes/相当于src目录-->

<init-param>

<!-- 这个param-name必须是contextConfigLocation-->

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/classes/applicationContext.xml</param-value>

</init-param>

<load-on-startup>2</load-on-startup>

</servlet>

　　<!--拦截所有以do结尾的请求-->

<servlet-mapping>

<servlet-name>dispatcherServlet</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

　<!--处理从页面传递中文到后台而出现的中文乱码问题-->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

每次配置好一个文件后 建议先启动服务器看看是否产生异常，不然到后期会很难调试和找到异常所在。

WEB.XML配置好以后 我们还需要在SRC目录下创建一个db-config.properties文件来存放我们的数据源配置信息：

内容如下：

db.url= jdbc:mysql:///springmvcdb?useUnicode=true&characterEncoding=utf8

db.username=root

db.password=root

db.dirverClass= com.mysql.jdbc.Driver

db-config.properties配置好以后就开始配置applicationContext.xml文件：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd"*>

<!-- 定义个缺省的控制适配器 -->

<bean

class=*"org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"* />

<!-- 获取配置文件 -->

<bean id=*"config"*

class=*"org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"*>

<property name=*"locations"*>

<list>

<value>classpath:db-config.properties</value>

</list>

</property>

</bean>

<!-- 获取数据源 -->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*>

<property name=*"driverClassName"*>

<value>${db.dirverClass}</value>

</property>

<property name=*"url"*>

<value>${db.url}</value>

</property>

<property name=*"username"*>

<value>${db.username}</value>

</property>

<property name=*"password"*>

<value>${db.password}</value>

</property>

</bean>

<!--

URL到处理器的映射列表可以配置多个

mappings属性健值为URL程序文件地址，而值为处理器的Bean名字，URL程序文件地址可采用路径匹配的模式，如：

com/mvc/t?st.jsp:匹配com/mvc/test.jsp、com/mvc/tast.jsp等 com/mvc /\*.jsp

：匹配所有com/mvc/下带jsp后缀的URL com/mvc

/\*\*/test.jsp：匹配所有在com/mvc路径下或子孙路径下的test.jsp com/mvc

/\*\*/\*.jsp：匹配所有com/mvc路径下或子孙路径下带.jsp后缀的URL cn/\*\*/web/bla.jsp：匹配

cn/option/web/dog.jsp cn/option/test/web/dog.jsp cn/web/ dog.jsp的请求

-->

<bean class=*"org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"*>

<property name=*"mappings"*>

<value>

user.do=userAction

</value>

</property>

</bean>

<!--定义视图 通过internalResourceView来表示 使用的是Servlet/jsp技术-->

<bean id=*"viewResolver"*

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"viewClass"*>

<value>org.springframework.web.servlet.view.InternalResourceView

</value>

</property>

<!--jsp存放的目录-->

<property name=*"prefix"*>

<value>/jsp/</value>

</property>

<!--jsp文件的后缀-->

<property name=*"suffix"*>

<value>.jsp</value>

</property>

</bean>

<bean id=*"userDao"* class=*"com.yjde.springmvc.UserDao "*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

<!--定义控制器-->

<bean id=*"userAction"* class=*"com.yjde.springmvc.UserController"*>

<property name=*"dao"*>

<ref bean=*"userDao"* />

</property>

<property name=*"commandClass"*>

<value>com.yjde.springmvc.UserDao</value>

</property>

<property name=*"viewpage"*>

<value>userInfo</value>

</property>

</bean>

</beans>

ApplicationContext.xml文件配置好以后我们开始编写具体的JAVA类

我们需要一个Dao类，一个controller类和一个PO

我们在oracle中创建了一张USERMBO表

里面有三个字段 USERID，USERNAME，USERAGE

UserDao类:

**package** com.yjde.springmvc;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.Collection;

**import** java.util.List;

**import** org.springframework.jdbc.core.RowMapper;

**import** org.springframework.jdbc.core.support.JdbcDaoSupport;

@SuppressWarnings("all")

**public** **class** UserDao **extends** JdbcDaoSupport {

**private** String msg;

**public** String getMsg() {

return msg;

}

**public** **void** setMsg(String msg) {

this.msg = msg;

}

//此方法把USEMBO表对应的字段查询出来依次放入userPO中

**public** Collection<UserPO> doquery() {

String sql = "SELECT T.USERID,T.USERNAME,T.USERAGE FROM USERMBO T";

**return** super.getJdbcTemplate().query(sql,new RowMapper(){

public Object mapRow(ResultSet rs, int num) **throws** SQLException {

UserPO user = **new** UserPO();

user.setUserId(rs.getInt("USERID"));

user.setUserName(rs.getString("USERNAME"));

user.setUserAge(rs.getInt("USERAGE"));

**return** user;

}

});

}

}

JdbcTemplate是core包的核心类。它替我们完成了资源的创建以及释放工作，从而简化了我们对JDBC的使用。 它还可以帮助我们避免一些常见的错误，比如忘记关闭数据库连接。具体请参阅API

Controller类：

**package** com.yjde.springmvc;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.json.JSONObject;

**import** org.springframework.validation.BindException;

**import** org.springframework.web.servlet.ModelAndView;

**import** org.springframework.web.servlet.mvc.SimpleFormController;

@SuppressWarnings("all")

//SimpleFormController是spring提供的表单控制器，把页面中Form中的元素名称设定为和bean中的一样，当传入的时候Spring会自动抓取form中和Bean名称一样的元素值，把它转换成一个bean,使得开发人员可以很方便的使用。

**public** **class** UserController **extends** SimpleFormController{

**private** String viewpage;

**private** UserDao dao;

**public** String getViewpage() {

**return** viewpage;

}

**public** void setViewpage(String viewpage) {

this.viewpage = viewpage;

}

@Override

**protected** ModelAndView onSubmit(HttpServletRequest request,

HttpServletResponse response, Object command, BindException errors)

**throws** Exception {

UserDao tmp = (UserDao)command;

Collection<UserPO> list = dao.doquery();

List<UserPO> users = **new** ArrayList<UserPO>();

UserPO user;

for (UserPO userPO : list) {

user= **new** UserPO();

user.setUserId(userPO.getUserId());

user.setUserName(userPO.getUserName());

user.setUserAge(userPO.getUserAge());

users.add(user);

}

Map mp = new HashMap();

mp.put("list", users);

**return** new ModelAndView(getViewpage(),mp);

}

public void setDao(UserDao dao) {

this.dao = dao;

}

}

UserPO类:

**package** com.yjde.springmvc;

**public** **class** UserPO {

**private** Integer userId;

**private** String userName;

**private** Integer userAge;

**public** Integer getUserId() {

**return** userId;

}

**public** **void** setUserId(Integer userId) {

**this**.userId = userId;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** Integer getUserAge() {

**return** userAge;

}

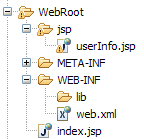
**public** **void** setUserAge(Integer userAge) {

**this**.userAge = userAge;

}

}

类创建完成以后我们编写两个JSP进行测试：



Index.jsp:

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<base href=*"*<%=basePath%>*"*>

<title>My JSP 'index.jsp' starting page</title>

<meta http-equiv=*"pragma"* content=*"no-cache"*>

<meta http-equiv=*"cache-control"* content=*"no-cache"*>

<meta http-equiv=*"expires"* content=*"0"*>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"This is my page"*>

<!--

<link rel="stylesheet" type="text/css" href="styles.css">

-->

</head>

<body>

<form action=*"user.do"* method=*"post"*>

请输入<input name=*"msg"* type=*"text"* />

<input type=*"submit"* value=*"提交"*>

</form>

</body>

</html>

userInfo.jsp:

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"utf-8"*%>

<%@page import=*"com.yjde.springmvc.UserPO"*%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

List<UserPO> str= (List<UserPO>)request.getAttribute("list");

%>

<html>

<head>

<base href=*"*<%=basePath%>*"*>

<title>SpringMVC</title>

<meta http-equiv=*"pragma"* content=*"no-cache"*>

<meta http-equiv=*"cache-control"* content=*"no-cache"*>

<meta http-equiv=*"expires"* content=*"0"*>

<meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>

<meta http-equiv=*"description"* content=*"This is my page"*>

<!--

<link rel="stylesheet" type="text/css" href="styles.css">

-->

</head>

<body>

<table border=*"1"* width=*"500"* align=*"center"*>

<tr>

<td >编号</td>

<td >姓名</td>

<td >年龄</td>

</tr>

<%

**for**(UserPO user:str){

%>

<tr>

<td><%=user.getUserId() %></td>

<td><%=user.getUserName() %></td>

<td><%=user.getUserAge() %></td>

</tr>

<%

}

%>

</table>

</body>

</html>

最终运行结果：



数据库建表语句：

/\*

Navicat MySQL Data Transfer

Source Server : mysql

Source Server Version : 50145

Source Host : localhost:3306

Source Database : springmvcdb

Target Server Type : MYSQL

Target Server Version : 50145

File Encoding : 65001

Date: 2012-04-24 10:34:25

\*/

SET FOREIGN\_KEY\_CHECKS=0;

-- ----------------------------

-- Table structure for `usermbo`

-- ----------------------------

DROP TABLE IF EXISTS `usermbo`;

CREATE TABLE `usermbo` (

`USERID` int(11) NOT NULL DEFAULT '0',

`USERNAME` varchar(50) DEFAULT NULL,

`USERAGE` int(11) DEFAULT NULL,

PRIMARY KEY (`USERID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of usermbo

-- ----------------------------

INSERT INTO `usermbo` VALUES ('1', '李晓红', '25');

INSERT INTO `usermbo` VALUES ('2', '柳梦璃', '27');

INSERT INTO `usermbo` VALUES ('3', '韩菱纱', '26');